Controller Area Networking

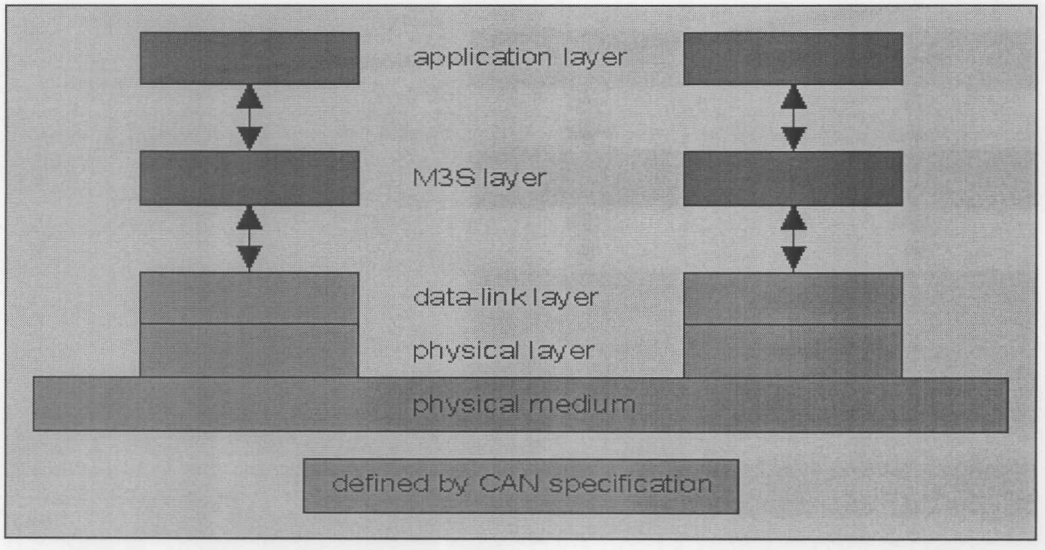
The Future Of Industrial Microprocessor Communications?

This article was released in the January 1995 issue of the ***Hitex UK Ltd.*** *C51/166 newsletter.*

HTML version by Olaf 'Olu' Pfeiffer, Hitex.

The CAN (Controller Area Network) is an ISO defined serial communications bus that was originally developed during the late 1980's for the automotive industry. Its basic design specification called for a high bit rate, high immunity to electrical interference and an ability to detect any errors produced. Not surprisingly due to these features the CAN serial communications bus has become widely used throughout the automotive, manufacturing and aerospace industries.

The CAN communications protocol describes the method by which information is passed between devices. It conforms to the Open Systems Interconnection model which is defined in terms of layers. Each layer in a device apparently communicates with the same layer in another device. Actual communication is between adjacent layers in each device and the devices are only connected by the physical medium via the physical layer of the model. The CAN architecture defines the lowest two layers of the model: the data link and physical layers. The application levels are linked to the physical medium by the layers of various emerging protocols, dedicated to particular industry areas plus any number of propriety schemes defined by individual CAN users. Perhaps the best example of an industry-standard CAN-based protocol is Allen-Bradley's DEVICEnetTM, designed for the networking of PLCs and intelligent sensors.



The physical medium consists of a twisted-pair with appropriate termination. In the basic CAN specification, it has a transmission rate of up to 250 kbaud whilst full CAN runs at 1MBaud.

The physical and data link layers will normally be transparent to the system designer and are included in any component that implements the CAN protocols. There are some microcontrollers with integral CAN interfaces, for example, the Philips 8051-compatible 8xC592 processor and the Siemens SABC167CR. The 80C200 is a standalone CAN controller which directly interfaces to many microcontrollers. The connection to the physical medium can be implemented with discrete components or with the 82C250 integrated circuit. Standalone CAN controllers are also available from Siemens, NEC and Intel.

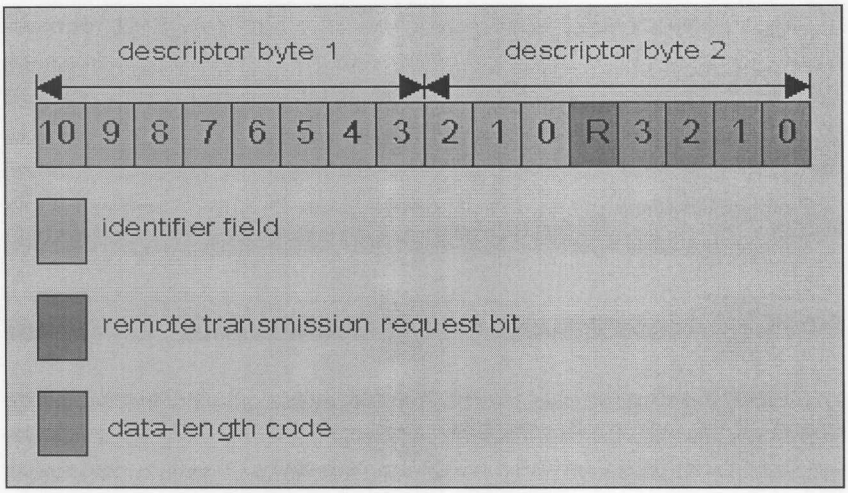
Low Cost Remote IO

Traditionally, CAN has been a network for coupling microcontroller-based devices meaning the cost per node is not particularly low. An interesting development is the concept of a "SLIO" module. This is a single chip which can act as a dumb input/output gateway on a CAN network, able to turn messages into real digital IO signals. It can also read IO pins and transmit the data as message plus use an integral A/D convertor to generate messages for introduction into a network. These devices are extremely cheap and are ideal for driving remote sensors, actuators or gathering digital and analog data. They can be viewed as remote add-ons to a central microcontroller. Currently, only basic CAN SLIOs are available but undoubtedly Siemens and other full CAN specialists will produce full CAN equivalents.

Two Varieties Of CAN

CAN has exists in two forms; a basic CAN and a higher form with an "acceptance filter". Basic CAN has a tight coupling between the CPU and the CAN controller, where all messages broadcast on the network have to be individually checked by the microcontroller. This results in the CPU being "tied up" checking messages rather than processing them, all of which tends to limit the practicable baud rate to 250kBaud. The introduction of an acceptance filter masks out the irrelevant messages, using identifiers (ID) and presents the CPU with only those messages that are of interest. This is usually referred to as "Full CAN". Philips is the leading proponent of basic CAN whilst Intel and Siemens only subscribe to full CAN. The Full CAN protocol allows for two lengths of identifiers: part A allows for 11 message identification bits, which yield 2032 different identifiers, whilst extended CAN (part B) has 29 identification bits, producing 536870912 separate identifiers.

Part A devices such as the Philips PCA82C200 are only able to transmit and receive standard CAN protocol. Used on an extended CAN system in which 29 bit IDs are present, the device will cause errors and crash the entire network. The Siemens 81C90 and 81C91 are similarly part A devices (11 ID bits), but have the ability to be used with extended CAN without causing no bus errors. This is achieved by simply ignoring the extended CAN frames and are thus known as "part B passive" devices. The data-link layer defines the format and timing protocol with which the messages are transmitted. There are two descriptor bytes and up to eight data bytes. The descriptor bytes are particularly important as they define the priority of the message and the type of message being transmitted.



The identifier field contains 11 bits and is used for identification of the message as well as for determining its bus access priority. The priority is defined to be highest for the smallest binary value of the identifier. The allocation of priorities to messages is a feature of the CAN bus that makes it particularly attractive for use within a strongly real time control environment. Bits 7-10 of the identifier field define the message priority. This means that messages can have a priority number between 0 (high priority) and 15 (low priority). The CAN specification guarantees the latency time associated with priority values, shown in table 1.

Coping With Message Collisions

As has been said, a fundamental CAN characteristic is that the lower the message number, the higher its priority - a identifier consisting entirely of zeros is therefore deemed to be the highest priority message. Thus if two nodes begin to transmit simultaneously, the first source to send a zero when the other source attempted a one, gets control of the CAN bus and goes on to complete its message. This use of "non-destructive bitwise" arbitration is coupled with the capability of each node to able to monitor its own transmissions. Thus if a transmitter 'A' is overruled by a source 'B' sending a higher priority message, the fact that the message read back does not match the message that 'A' attempted to send means that it will temporarily halt. Another attempt will subsequently be made to send it once the bus is released. This functionality is part of layer 1 and is contained entirely within the CAN controller device. It is therefore transparent to the CAN user.

Interactive Communication

It is possible to send a request for data to a specified address, and the remote transmission request (RTR) bit defines whether the message sent is a request for data or the actual data. The data-length code tells the receptor how many data bytes the message contains. In the case of data requests, no data bytes follow and therefore the data-length code has no direct relation to the number of data bytes.

The maximum number of nodes on a CAN bus is 32. The limit of messages per second ranges from about 2000 to about 5000 on a bus with 250kbaud transmission rate, depending on the number of bytes per message.

The Physical Layer

CAN can use a number of physical media such as twisted wire-pairs, fibre-optics etc. The commonest method is the former. The signalling is carried out using differential voltages and it is from this that CAN derives much of its noise immunity and fault tolerance. The two signal lines are termed 'CAN_H' and 'CAN_L' and in the quiescent state, sit at 2.5v. A '1' is denoted by CAN_H being above CAN_L and as such is termed a 'dominant' bit whilst zero has CAN_L above CAN_H, yielding a 'recessive' bit.

The use of voltage differentials allows CAN networks to function when one of the signalling lines is severed. Or in extremely noisy environments. With a simple twisted pair, the differential CAN inputs effectively cancel out noise, provided it is within the common mode range.

Cheap interfaces are available from Siliconix amongst others, which translate from 5v logic levels to the balanced line required by CAN.

Customising CAN To Specific Industries

The allocation of message numbers (and hence priorities) is up to the individual user but as has been indicated, certain industry groups are mutually agreeing the significance of certain messages and the exact protocol to be used. For example, manufacturers of motor drives might decide that message 0x10 is the winding current feedback signal from any motor on a CAN network and that 0x11 is the tacho speed. As 0x10 has a zero before 0x11, messages relating to current values will overrule those concerned with tacho readings.

In the case of DEVICEnet, PLCs from various manufacturers can be linked together. Peripheral devices such as pressure and temperature sensors can be added to the CAN network. As the messages generated by the sensors has been predefined, the PLCs will know that a certain message always relates to temperature, regardless of who actually manufactured it.

Situations Where CAN Is The Solution

CAN is ideal for any situation where microcontrollers need to communicate either with each other or with remote peripherals. In its home environment, the car, CAN was originally used to allow mission-critical real time control systems such as engine management systems and gearbox controls to exchange information. Here, CAN's short and guaranteed message latency times allowed each end of the network is working with current data, even where this may be changing on a hundreds of microsecond timescale. These systems all utilise full CAN in that the CAN controllers filter out unwanted messages to reduce the host CPU load. However, the appearance of low-cost standalone full CAN devices such as the Siemens 81C91 has allowed less time-critical tasks such as door systems (window lifters, mirror controls etc.) to become part of the CAN network. Indeed, the entire conventional wiring harness has been replaced in some instances by two-wire CAN networks in which even mundane devices such brake lights and indicators are just additional nodes.

In the meantime, basic CAN with 11 bit identifiers has become widely accepted as a general purpose network in the industrial control field. Developed and promoted mainly by Philips it allows very simple communication between microcontrollers and peripherals at up to 250kBaud. Indeed, the cheap SLIO device can provide up to 16 IO pins which may be assigned as up to 6 channels of 10-bit A/D or D/A, plus ordinary IO pins. SLIOs have unique identifiers, set via external resistors. Thus they can recognise messages intended for them and generate messages based on inputs received.

Industrial applications can also benefit from full CAN at 1MBaud by using full-CAN equipped microcontrollers from Siemens and Intel who also produce add-on CAN controllers for ordinary microcontrollers. However, the basic philosophy of full CAN is that it should be reserved for very high speed data interchange between microprocessing units rather than communication down to a low-speed IO port level.

CAN Support Tools

Despite its relative youth, CAN is already supported by a huge range of development tools. These range from simple development boards to full scale CAN analysers. Phytec of Mainz in Germany produce a wide range of CAN-equipped microcontroller boards based on the 8051 and SABC166 architectures. These comprise microcontroller + external CAN controller (Philips 82C200) or CAN microcontroller-only (80C592). The boards can be programmed in C or assembler and can host state-of-the-art debuggers such as Hitex's HiTOP. Phytec also offer small SLIO modules.

Softing of Munich produce a range of CAN tools and interfaces. These includeCAN-PCMCIA and VME slots plus a full CAN analyser, based on a conventional PC card or a miniture PCMCIA module - ideal for CAN debugging in the field.

Being so tightly coupled to microcontrollers, existing tools such as in-circuit emulators are able to provide useful facilities such as real time monitoring of input and output data to CAN controllers, whether on-chip or off-chip.

Controller Area Network (CANbus) --- June 4, 2000

Newsflash! The CAN one day seminar is now also available through FeabhaS, a vendor independent provider of training courses for Real-Time embedded system developers. FeahbaS courses are available as part of a public schedule - Or on-site.

The next public course in scheduled for July 12 near Reading, Berkshire, UK

Please see the FeabhaS web site for all the details.

The Controller Area Network (CAN)

- Is a high-integrity serial data communications bus for real-time applications
- Operates at data rates of up to 1 Megabits per second
- Has excellent error detection and confinement capabilities
- Was originally developed for use in cars
- Is now being used in many other industrial automation and control applications
- Is documented in ISO 11898 (for high speed applications) and ISO 11519-2 (for lower speed applications).

For more information - Click on one of the topics listed below.